WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, December 22, 2003

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L19	L18 and l12	10
☐	L18	L15	179
☐	L17	L16	0
		DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L16	L15	4
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L15	L14 and macro	183
☐	L14	(replac$ or transform$) near3 string	4783
☐	L13	((replac$ or convert$ or transform$ or chang$) near2 string)	10916
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L12	l11 or 712/300.ccls.	1403
☐	L11	717/110-118,141-143.ccls.	1104
		DB=JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L10	L9 and ((replac$ or convert$ or transform$ or chang$) near2 string)	5
☐	L9	(identif$ near (string or pattern))near2 (program or code)	127
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L8	L7	27
		DB=USPT,PGPB; PLUR=YES; OP=ADJ	
☐	L7	L6 and ((replac$ or convert$ or transform$ or chang$) near2 string)	34
☐	L6	(identif$ near (string or pattern))near2 (program or code)	399
		DB=JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L5	L4 and ((replac$ or convert$ or transform$ or chang$) near2 string)	5
☐	L4	(identif$ near (string or pattern))near2 (program or code)	127
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L3	L2	27
		DB=USPT,PGPB; PLUR=YES; OP=ADJ	
☐	L2	L1 and ((replac$ or convert$ or transform$ or chang$) near2 string)	34
☐	L1	(identif$ near (string or pattern))near2 (program or code)	399

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 10 of 10 returned.

☐ 1. Document ID: US 6651240 B1

L19: Entry 1 of 10 File: USPT Nov 18, 2003

US-PAT-NO: 6651240
DOCUMENT-IDENTIFIER: US 6651240 B1

TITLE: Object-oriented software development support apparatus and development support method

DATE-ISSUED: November 18, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yamamoto; Rieko	Kanagawa			JP
Nakayama; Yuko	Kanagawa			JP
Uehara; Tadahiro	Kanagawa			JP

US-CL-CURRENT: 717/108; 717/104, 717/116

ABSTRACT:

The object-oriented software development support apparatus according to the present invention includes: a pattern architecture storage unit for storing pattern information about a plurality of applicable patterns and inter-pattern relevant information for association among patterns having an antecedent/consequent relationship when the software is developed; and a pattern application unit for applying to a model to be developed a pattern whose application has been approved by the user among the patterns stored in the pattern architecture storage unit and detailing the model, and supporting the development of software corresponding to the model. The object-oriented software development support apparatus according to the present invention applies a pattern according to an entered pattern architecture, thereby enabling the pattern to be appropriately and efficiently reused, and efficiently supporting the development of software. In addition, various object-oriented patterns can be easily defined, and a model and a program desired by the developer can be automatically generated.

41 Claims, 85 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 85

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw. Desc | Ir

☐ 2. Document ID: US 6263493 B1

L19: Entry 2 of 10 File: USPT Jul 17, 2001

US-PAT-NO: 6263493
DOCUMENT-IDENTIFIER: US 6263493 B1

TITLE: Method and system for controlling the generation of program statements

DATE-ISSUED: July 17, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ehrman; John Robert	Sunnyvale	CA		

US-CL-CURRENT: 717/114; 717/141

ABSTRACT:

Disclosed is a system for processing program statements, such as statements included in a macro. An assembler program is provided a plurality of statements with an input file. The assembler program processes the statements. For each statement the assembler program determines whether the processed statement is a buffering directive including a statement operand. If the assembler program determines that the processed statement is a buffering directive, the assembler program writes the statement operand of the buffering directive into a memory area. Otherwise, the assembler program generates the processed statement into a data stream. The assembler program generates the statements stored in the memory area into the data stream after processing the statements associated with the macro.

32 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

☐ 3. Document ID: US 6081665 A

L19: Entry 3 of 10 File: USPT Jun 27, 2000

US-PAT-NO: 6081665
DOCUMENT-IDENTIFIER: US 6081665 A

TITLE: Method for efficient soft real-time execution of portable byte code computer programs

DATE-ISSUED: June 27, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Nilsen; Kelvin D.	Ames	IA		
Mitra; Simanta	Ames	IA		
Lee; Steven J.	Slater	IA		

US-CL-CURRENT: 717/116; 717/118, 717/148

ABSTRACT:

The invention is a method for use in executing portable virtual machine computer programs under real-time constraints. The invention includes a method for implementing a single abstract virtual machine execution stack with multiple independent stacks in order to improve the efficiency of distinguishing memory pointers from non-pointers. Further, the invention includes a method for rewriting certain of the virtual machine instructions into a new instruction set that more efficiently manipulates the multiple stacks. Additionally, using the multiple-stack technique to identify pointers on the run-time stack, the invention includes a method for performing efficient defragmenting real-time garbage collection using a mostly stationary technique. The invention also includes a method for efficiently mixing a combination of byte-code, native, and JIT-translated methods in the implementation of a particular task, where byte-code methods are represented in the instruction set of the virtual machine, native methods are written in a language like C and represented by native machine code, and JIT-translated methods result from automatic translation of byte-code methods into the native machine code of the host machine. Also included in the invention is a method to implement a real-time task dispatcher that supports arbitrary numbers of real-time task priorities given an underlying real-time operating system that supports at least three task priority levels. Finally, the invention includes a method to analyze and preconfigure virtual memory programs so that they can be stored in ROM memory prior to program.

89 Claims, 100 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 46

☐ 4. Document ID: US 6018628 A

L19: Entry 4 of 10 File: USPT Jan 25, 2000

US-PAT-NO: 6018628
DOCUMENT-IDENTIFIER: US 6018628 A

TITLE: Method of implementing parameterized types to be compatible with existing unparameterized libraries

DATE-ISSUED: January 25, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Stoutamire; David P.	San Juan Bautista	CA		

US-CL-CURRENT: 717/147; 709/331, 717/118, 717/141

ABSTRACT:

A method and apparatus for generating code using parameterized classes which is compatible with an existing class library that was previously generated using unparameterized classes is disclosed. According to the method, parameterized source code is received that contains variables that belong to a plurality of types which are

defined by supplying parameter values to a parameterized class definition. Static type checking is performed on the parameterized source code to determine if any incompatible type assignments exist between variables that belong to the plurality of types and values assigned to the variables. If no incompatible type assignments exist, then a homogeneous translation is performed on the parameterized source code to generate unparameterized class code. The unparameterized class code is then compiled to produce code that is compatible with the existing class library that was generated using unparameterized classes.

18 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

☐ 5. Document ID: US 6011916 A

L19: Entry 5 of 10 File: USPT Jan 4, 2000

US-PAT-NO: 6011916
DOCUMENT-IDENTIFIER: US 6011916 A

TITLE: Java I/O toolkit for applications and applets

DATE-ISSUED: January 4, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Moore; Victor S.	Boynton Beach	FL		
Walters; Glen R.	Sebring	FL		

US-CL-CURRENT: 717/104; 717/118

ABSTRACT:

A method to perform I/O (Input/Output) operations in a data processing unit running an interpretative based program in an Interpretive Machine (IM). One example on an interpretative based program running in an IM is a Java based program. An I/O class for passing data into the interpretative based program and out of a the interpretative based program is defined. An object representing an instance of this I/O class is created. The I/O class includes a first member function called an Applet function for handling I/O operations when the IM is running as an Applet coupled to a Web browser. The Applet function has its own procedures and data variables for performing I/O operations. The I/O class includes a second member function called and Application function for handling I/O operations when the IM is running as an Application not coupled to a Web browser. The Application function has its own procedures and variables for performing I/O operations. A check is made to determine whether the interpretative based program is being executed with or without a browser. When the interpretative based program is being executed with a browser, the Applet function performs I/O operations and when the interpretive based program is being executed without a browser, the Application function performs the I/O operations. In accordance with another embodiment of the present invention, a computer readable medium is disclosed corresponding to the above method.

17 Claims, 6 Drawing figures

Exemplary Claim Number: 1
Number of Drawing Sheets: 6

Full | Title | Citation | Front | Review | Classification | Date | Reference | [illegible] | [illegible] | Claims | KWIC | Draw Desc | Ir

---

☐ 6. Document ID: US 5999733 A

L19: Entry 6 of 10 File: USPT Dec 7, 1999

US-PAT-NO: 5999733
DOCUMENT-IDENTIFIER: US 5999733 A

TITLE: High speed assemble processing system

DATE-ISSUED: December 7, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Shamoto; Eiji	Kanagawa			JP

US-CL-CURRENT: 717/143; 711/221

ABSTRACT:

In an assemble processing system, when a syntactical analysis procedure syntactically analyzes a source program, a macro definition procedure stores a macro definition program body of the source program in a macro definition area, and a macro reference procedure stores a macro formal parameter and a macro local symbol of the source program in a symbol table. After the operation of the syntactical analysis means is completed, the macro reference procedure deletes the stored macro formal parameter and the macro local symbol from said symbol table.

8 Claims, 15 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 14

Full | Title | Citation | Front | Review | Classification | Date | Reference | [illegible] | [illegible] | Claims | KWIC | Draw Desc | Ir

---

☐ 7. Document ID: US 5754858 A

L19: Entry 7 of 10 File: USPT May 19, 1998

US-PAT-NO: 5754858
DOCUMENT-IDENTIFIER: US 5754858 A
**** See image for Certificate of Correction ****

TITLE: Customizable application project generation process and system

DATE-ISSUED: May 19, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Broman; David Michael	Redmond	WA		
DeMichillie; Leland Greg	Redmond	WA		

US-CL-CURRENT: 717/111

ABSTRACT:

Custom application project generators are created to generate specific types of computer application programs using an automated procedure implemented in a customizer tool. The customizer tool creates a custom generator project according to options chosen by a writer from a sequence of generator option selection steps. The custom generator project comprises source code files, templates, and dialogs which the writer can further modify using an editor. The custom generator project is compiled and linked to form a custom application project generator which implements an automated procedure for generating a specific type of application defined by the writer. The custom application project generator interfaces with a services module that provides default user interface and code generation services which can be overridden by the writer.

20 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

☐ 8. Document ID: US 4724521 A

L19: Entry 8 of 10 File: USPT Feb 9, 1988

US-PAT-NO: 4724521
DOCUMENT-IDENTIFIER: US 4724521 A

TITLE: Method for operating a local terminal to execute a downloaded application program

DATE-ISSUED: February 9, 1988

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Carron; James M.	Aiea	HI		
Uechi; Brian K.	Honolulu	HI		
Khan; Mohammed A.	Honolulu	HI		
Royston, III; Clifton W.	Honolulu	HI		
Abel; Jay A.	Honolulu	HI		
Ferlane; Bradley J.	Honolulu	HI		
Loui; Robert K. L.	Honolulu	HI		
Pape, III; William R.	Papaaloa	HI		

US-CL-CURRENT: 717/175; 379/91.01, 379/93.06, 379/93.17, 709/203, 709/219, 717/142, 717/176, 902/24, 902/37

ABSTRACT:

The present invention provides methods for operating a local terminal which includes a programmable computer so that the terminal executes a pre-arranged application program. More specifically, the present invention provides methods for operating a local terminal according to a pre-arranged application program which is created on a remote computer, then communicated by a communication channel to the local terminal where it is stored for execution.

18 Claims, 65 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 36

☐ 9. Document ID: US 4692896 A

L19: Entry 9 of 10 File: USPT Sep 8, 1987

US-PAT-NO: 4692896
DOCUMENT-IDENTIFIER: US 4692896 A

TITLE: Method of processing a plurality of code systems

DATE-ISSUED: September 8, 1987

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Sakoda; Kousuke	Hino			JP
Kainaga; Masahiro	Yokohama			JP
Akita; Hidehiko	Tokyo			JP
Murata; Fumiya	Hadano			JP
Nakaosa; Yoshitake	Yokohama			JP

US-CL-CURRENT: 717/142; 717/143

ABSTRACT:

A method of processing a plurality of different code systems for an information processing apparatus including an operating system, comprises a step of inputting a source program, and a compiling step of analyzing meaning of the source program to thereby create a series of instructions and data required for executing a processing equivalent to the meaning of the source program. The compiling step includes a sub-step of transforming character type constants to a first code system occupying region including a first number of bits and transforming character string constants to a second code system occupying a region including a number of bits which is equal to product of a sum of the number of characters of the character string constants plus one and multiplied with the first bit number so that character type variables designated in the source program correspond, respectively, to the region of the first bit number while a character type array corresponds to a region including a number of bits which is equal to a product resulting from multiplication of the first bit number with the number of elements of the array.

12 Claims, 20 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 14

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Ir

☐ 10. Document ID: US 4558413 A

L19: Entry 10 of 10 File: USPT Dec 10, 1985

US-PAT-NO: 4558413
DOCUMENT-IDENTIFIER: US 4558413 A

TITLE: Software version management system

DATE-ISSUED: December 10, 1985

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Schmidt; Eric E.	Los Altos	CA		
Lampson; Butler W.	Philadelphia	PA		

US-CL-CURRENT: 707/203; 717/110, 717/145, 717/171

ABSTRACT:

A software version management system, also called system modeller, provides for automatically collecting and recompiling updated versions of component software objects comprising a software program for operation on a plurality of personal computers coupled together in a distributed software environment via a local area network. The component software objects include the source and binary files for the software program, which stored in various different local and remote storage means through the environment. The component software objects are periodically updated, via a system editor, by various users at their personal computers and then stored in designated storage means. The management system includes models which are also objects. Each of the models is representative of the source versions of a particular component software object and contain object pointers including a unique name of the object, a unique identifier descriptive of the cronological updating of its current version, information as to an object's dependencies on other objects and a pathname representative of the residence storage means of the object. Means are provided in the system editor to notify the management system when any one of the objects is being edited by a user and the management system is responsive to such notification to track the edited objects and alter their respective models to the current version thereof.

6 Claims, 29 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 24

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Ir

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
L18 and L12	10

Display Format: - Change Format

Previous Page Next Page Go to Doc#

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 4 of 4 returned.

☐ 1. Document ID: JP 2001084147 A

L16: Entry 1 of 4 File: JPAB Mar 30, 2001

PUB-NO: JP02001084147A
DOCUMENT-IDENTIFIER: JP 2001084147 A
TITLE: DEVICE AND METHOD FOR CONVERTING CHARACTER STRING, AND RECORDING MEDIUM

PUBN-DATE: March 30, 2001

INVENTOR-INFORMATION:

NAME	COUNTRY
MOTOYAMA, TETSURO	
YEVUGEENYA, RYAPUSUTEIINA	

INT-CL (IPC): G06 F 9/45; G06 F 5/00

ABSTRACT:

PROBLEM TO BE SOLVED: To convert encoded visual information from one language to another language by replacing a character string to macro related with the character string and generating and storing an entry in a map using the relation of the macro and the character string.

SOLUTION: For example, an input file 102 includes a set of computer instructions having an encoded character string 118 having a value abc. In order to convert character strings 118 to 126, a converting mechanism 108 replaces one corresponding character string with a corresponding macro character string. The mechanism 108 copies respective input files 102 to 106 to corresponding output files 112 to 116. In the middle of copying, the mechanism 108 generates and stores a map table having an entry for assigning only one macro character string to each characteristic character string identified in the unit of a code. The character string 118 is replaced with the only one macro character string (Z0001) 140.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Ir

☐ 2. Document ID: NN9201125

L16: Entry 2 of 4 File: TDBD Jan 1, 1992

TDB-ACC-NO: NN9201125

DISCLOSURE TITLE: International Message Support for Aix Nroff/ Troff Macro Packages.

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, January 1992, US

VOLUME NUMBER: 34
ISSUE NUMBER: 8
PAGE NUMBER: 125 - 126

DISCLOSURE TEXT:

- Disclosed is a software solution for integrating the nroff/troff macro packages with the AIX* International Message Facility. - Traditionally, the nroff/troff macro packages use static English text messages for such things as page headers, day and month names, etc. A requirement for the internationalization of programs is that text displayed on the output devices be in the language of the user. Using the traditional method requires each macro package to be translated in its entirety and shipped as a locale-specific package. Furthermore, it requires the maintenance of multiple, almost identical macro package source files. - Described is a solution for internationalization of the macro packages that saves resources. First, all static text messages are externalized into message catalogs. Secondly, code changes to the nroff and troff commands allow the string manipulating directives to access the message catalogs to retrieve the translated text. The macro packages are maintained separate from the translated text messages, thereby requiring only one set of macro packages to be maintained and shipped worldwide. The translated text messages are packaged as part of the base system messages for each specific locale. - The nroff and troff commands have three directives which directly manipulate text or write text to the output devices. ***** SEE ORIGINAL DOCUMENT ***** To interface with the message catalogs requires the use of a second syntax for each directive. The second syntax has a minimum of a set number and message number identifier. The alternate syntax may contain a default message which is used if the message retrieval fails. The order of some words may differ between languages; therefore, the alternate syntax uses variable arguments that can be ordered in the output by replaceable format strings in the message text. - To maintain backward compatibility with the traditional syntax, control characters are used to identify the alternate syntax method. The control characters are also used to delimit the components of the alternate syntax. The ASCII code Control-A (areA) delimits message identification, default message and the optional argument list. The ASCII code Control-B (areB) delimits individual optional arguments. 2 is the message set number. 41 is the message number. "..." text within quotes is the default message. \n(.F is the name of the current input file. \n(.c is the number of lines read from input file. If the troff command runs with these conditions: - The message at set 2 and number 41 matches the default message. - The current input file is paper.doc - The .ds directive is on line 124 in the input file. - An English locale then message identified by 2 and 41 is retrieved from the catalog; the replaceable format strings are set with the variable arguments and the string {c would be defined as: ERROR: (paper.doc) input line 123 The example repeated in a different locale using the same macro package source will define a message in the language of that locale.

☐ 3. Document ID: NN9004146

L16: Entry 3 of 4 File: TDBD Apr 1, 1990

TDB-ACC-NO: NN9004146

DISCLOSURE TITLE: IBM PC/3270 Printer Definition File.

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, April 1990, US

VOLUME NUMBER: 32
ISSUE NUMBER: 11
PAGE NUMBER: 146 - 147

DISCLOSURE TEXT:

- The IBM PC/3270 Host Directed Print Program supports 13 IBM Personal Computer printers. Each printer has some of its own unique features. The problem is how to tell the executable code how to perform certain actions on a given printer. This is accomplished by the provision of a Printer Definition File which is designed to contain all essential information about a printer to be used with PC/3270 Host Directed Print. There is one such file per supported IBM printer. The Printer Definition File is the input to the Printer Table Compiler, which together with other PC/3270 utilities converts this file into a table for use by the executable code. The PC/3270 software is able to support future printers without code changes by simply creating Printer Definition Files for these printers. - The Printer Definition File has four main sections. 1. Macro Definition This section is enclosed by two keywords: BEGIN MACRO and END MACRO. Each macro definition consists of three parts: a. Name: The name must be a string of three or more characters, and begins with an alpha. b. EQU: This is a key word, and must appear following the name. c. Value: Value is a string of hex numbers that defines the name. Once defined, the macro names can be used on the right side of the definitions in the other three sections. During processing, the name will be replaced by the string of hex numbers defined in this section. 2. Session Parameters This section describes the session specific parameters such as maximum page length and maximum print positions. 3. Control Codes This section describes how controls such as backspace, set horizontal tab, and color printing, can be performed on this printer. 4. Character Definitions This section describes how each of the 192 EBCDIC graphic characters can be printed on this printer. This sections serves two purposes: 1. It serves as Code Page translate table. Note that each graphic character is given a name. It can be defined as a single hex number if the printer can print that character directly. 2. A character that cannot be printed directly can be defined a substitute, or may be emulated by bit map printing or by overlay printing. - By using the Printer Definition File to define the printer characteristics to the executable code, the PC/3270 software is able to support future printers by creating new Printer Definition Files for them without code change.

☐ 4. Document ID: RD 309002 A

L16: Entry 4 of 4 File: DWPI Jan 10, 1990

DERWENT-ACC-NO: 1990-056468
DERWENT-WEEK: 199008

TITLE: DSL constant preprocessor for DPP replacement - has logic design description language preprocessed into several modern programming constructs not available in base language

PRIORITY-DATA: 1990RD-0309002 (January 20, 1990)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
RD 309002 A	January 10, 1990		001	

INT-CL (IPC): G06F 0/01

ABSTRACTED-PUB-NO: RD 309002A
BASIC-ABSTRACT:

Constant values for DSL net names can be added at the top of the 18L source. DPP will find all uses of the predefined constant nets and replace the name with the constant number (essentially similar to programming language macro preprocessors). The replacements are done with string replacements so other uses of the replacement capability are possible. Replacements are also done inside of constant definitions so nesting of definitions is supported.

DSL forces all case statement labels to be absolute literals such as B'0000110'. The DPP replacement allows a designer to use named labels that greatly increase the readability, flexibility, and accuracy of these statements. The preprocessor creates source code with ''LISTING'' information in the DPP generated comments. All DPP definitions are commented out and all replacements are tagged with the name of the constant in a comment. After the DPP string replacement, the source code line may be over the 72 character limit imposed by DSL. DPP will break these lines into multiple new lines to provide completely compilable source code.

ADVANTAGE - Improve readability and reduced risk of designer errors.

Terms	Documents
L15	4